The Rubber Meets the Road: Integrating the Unified Medical Language System Knowledge Source Server into the Computer-based Patient Record

Kevin B. Johnson, M.D., M.S.,[§] and Edwin B. George, M.D., Ph.D.[†]

[§] Division of Biomedical Information Sciences, The Johns Hopkins University School of Medicine, Baltimore, Maryland, and [†]Department of Neurology, Wayne State University, Detroit, Michigan

Ongoing improvements in the content of the Unified Medical Language System, coupled with the recent release of the Internet-based Knowledge Source Server (KSS), have prompted us to develop an interface between the KSS and our computer-based patient record. We confronted many challenges while developing a robust interface to an Internet-based server, and in integrating the process of codification into the workflow of clinicians. An initial evaluation of the interface in the clinical environment suggests that acceptable performance is attainable. The benefits of using an Internet-accessible tool in a clinical information system appear to justify the effort required.

INTRODUCTION

The National Library of Medicine's Unified Medical Language System (UMLS) project is an extensive international research and development effort aimed at creating a tool that can facilitate the coding of biomedical information, and providing a mechanism to share this information among disparate systems.[5] Through yearly updates, UMLS has become a more robust representation of biomedicine. Most recently, the tool has incorporated the SNOMED International system for disease classification—a step that is critical for UMLS to be considered a comprehensive representation of clinical problems and diagnoses. [2,6-9]

These changes to the UMLS enhance its utility as a tool for codifying patient care information. An essential component of the medical record that can greatly benefit from codification is the patient's problem list. This list is formulated or updated by the physician at each clinical encounter, and has been a challenge to codify at the point of care.[3,7,10-13] The most success to date has been demonstrated using UMLS and SNOMED. Campbell and colleagues have suggested that a combination of SNOMED and UMLS may offer "different and complementary advantages" [9]—an environment that is now realized through recent enhancements to the UMLS.

In 1995, the National Library of Medicine (NLM) made UMLS available through an Internet-based Knowledge Source Server. [1,4] The Knowledge Source Server (KSS) can be accessed in three ways, including access through NLM-developed programs that allow workstations with access to the Internet to submit command language queries to the KSS. This program code can be used as an Application Programming Interface (API) to allow application programs to access the Internet-based KSS in real time.

Providing an API to the KSS offers many benefits to application developers and NLM staff. The API greatly simplifies the task for developers interested in accessing and manipulating UMLS data. In addition, with the API, end-user applications also will be shielded from minor changes to the server programs that access the UMLS. Of course, use of a central UMLS server simplifies the NLM's distribution of the knowledge sources and might allow more effort to be spent updating the KSS more frequently.

We are conducting a 2 phase project utilizing the KSS to codify patient medical problems at the time of data-entry by our physician staff. We are examining the resulting problem lists to determine their usefulness in identifying specific subgroups of patients. The goal of our first phase was to integrate the UMLS tools into our computer-based patient record (CPR), using the KSS. We report here our observations regarding this phase of the project.

DESIGN CONSIDERATIONS

In developing a plan to integrate UMLS query capability into our existing problem list entry tools, we had four major objectives:

1. Identify any impediments to matching UMLS terms to the physician-identified patient problems.
2. Modify our user interface to support codifying physician-identified patient problems using the KSS, while ensuring ease of use, and with respect to issues identified through Objective 1.
3. Develop a robust and stable interface between our CPR and the KSS. Identify any impediments to the development process.
4. Ensure that the system works well on a variety of workstation environments, as is commonly encountered in decentralized medical institutions.

1091—8280/97/$5.00

Impediments to Term Matching

Our CPR is designed to provide maximum flexibility for the clinician-user trying to enter clinical information. This system has been in place for 2 years, and is used by most of our physician staff. To date, our staff have entered an average of 3.3 problems for each of the 3902 patients in the CPR. Entering a medical problem requires typing in text describing the problem, and modifying the date of onset if necessary.

In contrast to the system described by Zelingher, we do not supply a text field for comments. Rather, we allow these phrases as a part of the problem description, so that all relevant modifiers are apparent upon inspection of the problem list. While this design creates an intuitive data-entry tool, it can confound the process of matching user input to a UMLS term. We analyzed our existing problem list, using as an example all patients with a diagnosis of asthma. From our problem list, we identified 193 patients with asthma and categorized the contents of their problem text, as shown in Table 1. Within the text, there were spelling errors, disease severity indicators (e.g., "mild"), critical descriptive information (e.g., "cough-variant"), other descriptive information (e.g., "diagnosed by [a pulmonologist]"), and uncertainty indicators (e.g., "probable"). Inspection of the database contents also disclosed temporal modifiers (e.g., "history of ") and compound problems (e.g., "viral URI with otitis media"). In addition, we found 13 different terms for "asthma" (e.g., "RADE", "status asthmaticus"), comprising 55% of the descriptors for asthma. These aliases pose a significant challenge to categorizing patient problems.

Category	Occurrences (%)
Spelling errors	3 (2)
Disease severity indicators	40 (21)
Descriptive information	14 (7)
Diagnostic uncertainty	5 (3)
Aliases for "asthma"	106 (55)

Table 1. Analysis of problem list entries for patients with asthma (n = 193)

User Interface Development

Our goal for the UMLS version of the problem entry screen was to minimize the disruption caused by the coding process, and also to leverage the training and "cultural" change that had already occurred among our physicians. Optimally, the system would silently match the text string input by the clinician-user to a UMLS term, and invisibly store the associated Unique Concept Identifier (CUI). However, we recognized that the user interface had to support not only the process of submitting a text string the KSS, but also clinician-user response to a variety of possible outcome scenarios occurring when a term match did not occur.

Scenario #1: many possible matches, including lexical variants. In most instances, the clinician-user will need to choose from among a set of text strings returned from the KSS as possible matches. These text strings can have subtle, but sometimes clinically significant, differences. The interface needs to provide feedback that seduces the user into choosing the most clinically relevant expression.

Scenario #2: no matching UMLS terms. There are a variety of ways that a text string can fail to match a UMLS term. In addition to the obvious failure to code these problems, we have noticed that when we have submitted strings with misspellings, we have had to endure a long wait (sometimes as much as 15-30 seconds) to discover that there was no match. Our user interface needs to facilitate the entry of problems even when no matching UMLS term is found.

Scenario #3: multiple concepts to describe one user problem. Problems that contain anatomic information as well as a disease entity form one cogent example of a common challenge we needed to overcome. For example, two specific concepts are needed to codify the content of the problem *temporal lobe stroke*. Anatomic information is an essential component of many diagnoses and problems; the interface needs to help the user work through this scenario.

API Development

Our initial testing of the code provided by the NLM disclosed an architectural approach that proved less reliable in our environment. The NLM code relied on synchronous network database and socket functions, using iterative loops to avoid permanently blocking while waiting for server responses over the network. This approach is not robust in the non-preemptive multitasking environment of 16-bit Windows™, which remains the most predominant desktop workstation configuration in our institution at this time. In 16-bit Windows, synchronous Winsock™ calls monopolize processor time resulting in unacceptable system performance, especially when network delays are encountered.

Workstation environment considerations

One consistency within most medical organizations is the lack of a standard, ubiquitous configuration for desktop workstations. In our environment, CPR users work on 16 and 32 bit Windows™ environments, using a range of microprocessors. A few users run Windows-emulation on the Macintosh™ platform. Early in our system testing, we discovered differences in memory management between critical dynamic link libraries (DLLs) on the various platforms that significantly compromised the performance of the application.

Therefore, as with other desktop workstation applications, it became essential to test the UMLS interface to our CPR in all settings every time changes were made to any part of the program.

Other CPR Modifications

Clinical systems must provide very robust performance, and "fault-tolerant" behavior is a necessity. In any Internet-based application, the design of the system must take into consideration a variety of unpredictable events, including: network and hardware malfunctions, slow network throughput, operating system errors, and remote server downtime. Any of these conditions, which confront users of the World Wide Web on a frequent basis, should have no effect on users in busy clinical environments. Therefore, we need to provide robust mechanisms to recover from problems that affect system performance.

SYSTEM DESCRIPTION

The changes to our CPR resulted in a new dynamic link library (DLL), a new user interface, and some changes to the architecture of our CPR to support these modules.

The KSS API

We elected to encapsulate the API to the KSS in a 16-bit Windows™ Dynamic Link Library (DLL). This library provides a variety of functions including those in the original NLM code. The DLL then uses asynchronous Winsock operations to access the KSS to post and retrieve the query results, yielding CPU time as needed on behalf of the calling application. The DLL uses buffers, windows and a message queue allocated on behalf of the calling application to handle multiple calls from different applications or possible reentrant execution. This API helps to isolate the production application code from changes to the UMLS Knowledge Sources Server, changes in the network interface, and variations in the application platform.

There are exported functions for Metathesaurus queries, Semantic Network queries, Specialist Lexicon queries, and Information Sources Map queries. In addition, there is a generic query function which takes one of these services as a separate parameter, which provides for any future expansion in service types. In addition, the DLL also includes two specialized query functions, PickListQuery and GetConcept, provided specifically for interaction with our interface module.

Four additional management functions have been added. A display function uses an integer flag to determine whether the DLL should perform queries silently in the background. There also are display attributes to support the debugging of applications that use the DLL. A timer function allows the calling application to set a time-out value to determine how long a query should wait for a response from server before returning with a partial or no response. The DLL communicates with an initialization file to store such items as the Internet Protocol address of the Knowledge Sources Server, and other parameters that affect its performance. A third management function allows the calling application to reinitialize the DLL from the default initialization file or from an alternative initialization file specified by the user. The fourth management function allows the application calling the DLL to obtain a descriptive string associated with error codes returned by any DLL query function. Together, the functions exported by the DLL provide a simple but powerful and expandable API which isolates the calling applications from network operations while providing the full functionality of the KSS in all Windows™ environments.

User Interface

The user interface module has been constructed using Visual Basic™. The interface is shown in Figure 1. The interface is designed to be intuitive, and to closely approximate the functionality of our free-text problem entry interface. A large text field receives a problem list entry from the user. When the user presses the UMLS SUBMIT button, the PickListQuery function searches the KSS to generate a set of UMLS terms that are sorted and placed in a pick list. The user may then select an item in the pick list or select from a variety of other actions, including expanding the pick list with normalized strings for other related concepts. If the clinician selects an item from one of the pick lists, the text is replaced by the selected term and the associated concept identifier (CUI) is stored. The clinician may make additional changes to the text string before saving it in the patient problem list. If the UMLS query fails to match any terms, or if the clinician selects the "NONE OF THE ABOVE" option on the pick list, the original text string is stored with an identifier indicating an unknown concept.

For the purposes of analysis, the interface creates a record of every user interaction, including what text strings the user entered, use of the expand, restart and cancel buttons, total time expended and idle time during the interaction. Idle time is when the window or child windows of the module do not have the focus, or when a window of the module does have the focus but there is no keyboard activity or mouse clicks for longer than 10 seconds. This record is stored in a separate database for future analysis of the user interaction with the interface.

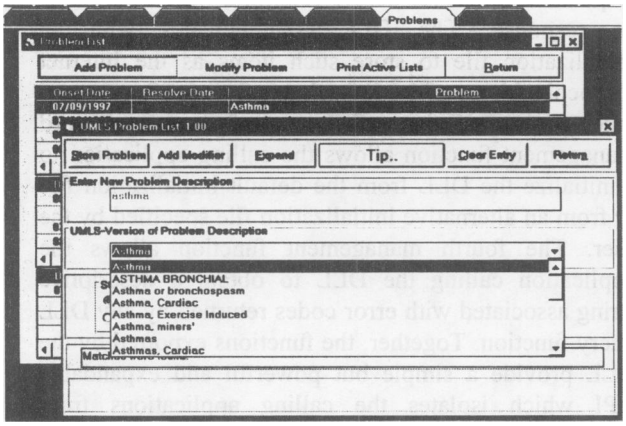

Figure 1. View of UMLS Problem-entry screen. After submitting the phrase "asthma" to the KSS, the system displays a list of strings that match the clinician input. The clinician chooses "asthma, exercise induced" then presses the "STORE PROBLEM" button to store this codified problem into our production database.

Preprocessing User Input

Initial testing of the UMLS queries with clinician-entered text from the CPR disclosed potential sources of poor term matching that we could avoid by preprocessing the text. The most common problems were caused by modifiers placed in front of the actual problem text. The user interface was modified to inspect the string for the occurrence of any of these phrases before any string is sent to the PickListQuery function. If any modifiers are found, they are removed from the user string and then reinserted in the user string ,if an exact match to the remainder of the string is found. Otherwise, the user is able to reinsert these words after selecting a phrase from the pick list. Included among the modifiers that we remove are "s/p", with a frequency of 4.5% in our list, and "hx of", with a frequency of 6.2%.

User Feedback When No Match Is Found

Our initial testing confirmed our suspicions about the need to handle patient problems that contain two separate UMLS terms. In addition, we encountered other reasons for a user string to not match a normalized string.

One common situation occurs when a one-word phrase fails to match any UMLS term, which is usually the result of an abbreviation or misspelling. When a single word does not match any terms, a button labeled "TIPS" begins to blink. If the user selects this button, the system suggests that the user verify the spelling and remove any abbreviations before resubmitting the text to the KSS. The pick list also displays the initial string as the only option. If this is selected, it will be stored, under a concept identifier to denote that there was no matching UMLS term.

Occasionally, a user might type in a phrase that does not match a UMLS term. If there is no match, we see if the phrase can be decomposed into more than one unique concept, by submitting each word to the KSS and examining the result. If the phrase matches one or more concepts, then the user is given an option to select one, the other, or both phrases to compose their problem. We store up to two concepts (CUIs) for each problem in our problem list. If neither word generates a match, the "TIPS" button blinks to suggest using less words to define the problem. The initial string is the only option in the pick list.

LIMITATIONS

Our testing of the API and our user interface has been reassuring. However we have encountered some limitations both in our design and in the API toolset.

The current KSS does not provide a capability for managing misspelled words. At present, we do not perform a spell check before submitting words to the UMLS. Verifying the spelling would greatly speed up the process of codification.

One significant limitation to our interface is that we do not currently display concept information about any UMLS term. In most cases, this omission does not result in misclassification. However, there are examples of strings that match more than one concept, such as "cold", and strings that match only one concept, but have multiple meanings, such as "RAD" which is a radiographic term that is contained within the UMLS, but which also is an abbreviation for reactive airways disease. We have not provided the clinician with tools necessary to resolve either of these ambiguities in our interface. More work will need to be done to develop a user interface that can provide better feedback about the concepts associated with a particular string.

USER REVIEW OF THE UMLS INTERFACE

After completing the design and implementation of our user interface, we obtained feedback from a group of nurses, housestaff, and attending physicians. This focus group made the following observations:

- The interface is similar to the existing user interface for creating problem list entries
- The TIPS button is an extremely useful and clever tool for providing feed-back and changing user behavior in an unobtrusive way
- UMLS found a match for most common problems entered by members of the focus group
- The time from submitting a text string to obtaining a pick list result may be too long

- The ability to combine two concepts into one patient problem is not intuitive

As mentioned by our users, the latency introduced by sending a query to the KSS to generate the pick list remains a major limitation in the system. Often, transmission delays due to network congestion are a significant component of this latency. The mean response time recorded in our research database is currently 13 seconds; however the mode is 3 seconds and the median time is 8 seconds. Experience shows that there is a latency component at the KSS which decreases as users improve their strategy in submitting text for matching. Thus, we expect this mean response time to improve as more of our users gain experience with the UMLS-enabled CPR.

CONCLUSIONS

We have successfully deployed a problem list entry mechanism that integrates access to the Internet-based UMLS KSS during the time of data entry. There were significant challenges in designing a user interface which minimizes the impact of the coding process on the efficiency of the resident and attending staff. Further analysis is being conducted to determine the effect this interface has had on the behavior of our staff, as well as the effects it has on the usefulness of our patient problem list. Despite the challenges we encountered, many of which were previously known difficulties of employing an encoding tool such as UMLS or previously known difficulties in distributed computing, we are very encouraged by the outcome of our project to date.

Acknowledgments

This project was supported by order number 467-MZ-502170 from the National Library of Medicine. We wish to thank Delores Allen, Alan Coltri, Gani Cortez, Lisa Elliott, Harold Lehmann, Ronald Lesser, and Stephanie Reel for their assistance and support.

References

1. McCray AT, Razi AM, Bangalore AK, Browne AC, Stavri PZ. The UMLS Knowledge Source Server: A Versatile Internet-Based Research Tool. 1996 AMIA Annual Fall Symposium 1996: 164-168.

2. Chute CG, Cohn SP, Campbell KE, Oliver DE, Campbell JR. The content coverage of clinical classifications. For The Computer-Based Patient Record Institute's Work Group on Codes & Structures. J Am Med Inform Assoc 1996; 3:224-233.

3. Linnarsson R, Nordgren K. A shared computer-based problem-oriented patient record for the primary care team. Medinfo 1995; 8 Pt 2:1663

4. McCray AT, Razi A. The UMLS Knowledge Source server. Medinfo 1995; 8 Pt 1:144-147.

5. Lindberg DA, Humphreys BL, McCray AT. The Unified Medical Language System. Methods Inf Med 1993; 32:281-291.

6. Campbell KE, Musen MA. Representation of clinical data using SNOMED III and conceptual graphs. Proc Annual Symp Comput Appl Med Care 1992; 354-358.

7. Payne TH, Martin DR. How useful is the UMLS Metathesaurus in developing a controlled vocabulary for an automated problem list? Proc Annu Symp Comput Appl Med Care 1993; 705-709.

8. Warren JJ, Campbell JR, Palandri MK, Stoupa RA. Analysis of three coding schemes: can they capture nursing care plan concepts? Proc Annu Symp Comput Appl Med Care 1994; 962

9. Campbell JR, Payne TH. A comparison of four schemes for codification of problem lists. Proc Annu Symp Comput Appl Med Care 1994; 201-205.

10. Wilton R. Non-categorical problem lists in a primary-care information system. Proc Annu Symp Comput Appl Med Care 1991; 823-827.

11. Campbell JR, Payne TH. A comparison of four schemes for codification of problem lists. Proc Annu Symp Comput Appl Med Care 1994; 201-205.

12. Scherpbier HJ, Abrams RS, Roth DH, Hail JJ. A simple approach to physician entry of patient problem list. Proc Annu Symp Comput Appl Med Care 1994; 206-210.

13. Zelingher J, Rind DM, Caraballo E, Tuttle MS, Olson NE, Safran C. Categorization of free-text problem lists: an effective method of capturing clinical data. Proc Annu Symp Comput Appl Med Care 1995; 416-420.